Un Ejemplo de Aplicación de la Tecnica Bayesiana y Razonamiento Basado en Casos en el juego del fútbol.

Lic. Ariel González
Departamento de Computación
Facultad de Cs. Exactas, Físico – Químicas y Naturales
Universidad Nacional de Río Cuarto
Córdoba - Argentina
gonzalezg@exa.unrc.edu.ar
(054) 0358 - 4676235 / 529

Resumen

En el presente artículo se describe una propuesta de solución al problema del dominio del fútbol, la cual resuelve el problema de las acciones que un jugador de fútbol debería realizar, mediante la integración cooperativa de las Redes Bayesianas ([7] y [8]) y el Razonamiento Basado en Casos [1]. Esto incluye las tareas dinámicas de dos equipos, y este artículo se concentra en el fútbol simulado como un ejemplo. Primero, se analizan cuales son los elementos del problema en cuestión, en base a ellos se proponen distintos sensores para obtener información de un jugador y de los objetos de un campo de juego. Por último se presenta un set acciones abstractas que un jugador podría realizar.
Se utilizan las redes Bayesianas para caracterizar la selección de una acción donde el método Razonamiento Basado en Casos es usado para determinar cómo llevar a cabo tales acciones (ambos temas son tratados en conjunto pero con una visión diferente en [6]).

Palabras Claves:
Redes Bayesianas, Razonamiento Basado en Casos, Simulación de Jugador de Fútbol.

Cantidad de Palabras: 3.561

Tema:

"Inteligencia Artificial"

Actividad:

"Workshop de Agentes y Sistemas Inteligentes"

Introducción

En este artículo se ejemplifica las tareas que un jugador del fútbol (robot) debería llevar a cabo bajo ciertas situaciones, algunas de ellas fueron extraídas de [10]. El jugador del fútbol es representado por un conjunto de objetos y funciones en los objetos. El idioma del problema está basado en un simulador de jugador de fútbol "RoboCup" que incluye estados, acciones, y objetos.
Las acciones son funciones que cambian el estado del ambiente y los estados son representados en base a entradas sensoriadas primitivas que se describen en las siguientes secciones.

Los objetos

Los objetos en nuestro dominio son "móviles" (es decir, la pelota y el jugador) o "estáticos" (ej., posiciones del campo de juego, las líneas del campo de juego, y los arcos).
La siguiente tabla muestra las abreviaciones que se usarán para representar los objetos.

Objeto	Abreviación
Cualquier Objeto	*O*
Objeto Movil	*M*
Jugador	*A, D* (es decir, atacantes y defensores)
Pelota	*b*
Posición	*p*
Arco	*g*

Los sensores

El ambiente se describe por un juego de entradas sensorias. Se dispone de cuatro sensores:

***orientation*(A)**: da la dirección que el jugador **A** lleva actualmente. El juego de posible direcciones es: {N;NE;E; SE;S;SW;W;NW}.

***position*(O)**: da la posición bi-direccional de el objeto **O**.

***team*(A)**: da el equipo que el jugador **A** es afiliado.

***velocity*(M)**: da la velocidad de objeto móvil **M**.

Las acciones que se pueden seleccionar, serán en base a la información que estos sensores arrojen. Sin embargo, dado que estos sensores primitivos pueden ser imprecisos, funciones más abstractas basadas en ellos son útiles para resumir el ambiente en un idioma más útil para la fabricación de decisiones. Estas funciones son:

distance(p1, p2): da la distancia Euclideana entre la posición *p1* y la posición *p2*.
near(O1,O2): los objetos *O1* y *O2* están cercanos con certeza 1 si ellos están en posiciones adyacentes. La creencia de un jugador que este predicado es verdad es definida como la probabilidad que O1 puede estar cercano a O2 dentro de algún intervalo tiempo finito[1].

[1] Este intervalo de tiempo, estimado por A1, es el tiempo esperado necesitado para que la pelota viaje de A1 a A2, cuando A1 está pasando la pelota a A2. La creencia de un jugador en la verdad de algunos predicados debe evaluarse con respecto a una duración de tiempo. Para ahora, se asume que estos aspectos temporales son incluidos, pero se ignorarán en el resto de este artículo para simplificar la discusión inicial.

near_line(O1,O2,O3): el Objeto *O1* está cercano a la línea de segmento que une a los objetos *O2* y *O3*, si existe un punto *p* en ese segmento tal que *near(O1,O4)* es verdad para cualquier objeto O4 con *position(O4, p)*.
open(O): el Objeto *O* está abierto si no existe ningún (otro) jugador *A* tal que *near(O,A)*.
teammates(A1, A2): es verdadera sii estos jugadores son del mismo equipo.
intercetp_threat(D,A,p): es verdadera sii existe un defensor *D* tal que ¬*teammates(D,A)*, *A* está intentando patear la pelota hacia un compañero *A2,* y existe una posición *p* en la línea de segmento entre *A1* y *A2* tal que *near(D, p)*.
goal(A,g): da probabilidad de que el jugador *A* anote un gol en el arco *g*, suponiendo que *open(A)* y además existe otro jugador D tal que *Intercetp_threat (D, A, position(g))* (Arquero).
possession(A): verdadero si el jugador *A* tiene la pelota y no hay ningún jugador D tal que *near(D, A)*.
relative_orientation(O1,O2, d): da la orientación relativa de objeto *O2* desde el objeto *O1* respecto a una dirección *d*.
visible(A,O): es verdadera sii la dirección relativa del objeto *O* desde *A* está dentro de 90 grados de orientation(A) (recordar el esta ultima función es un sensor que da la dirección que lleva el objeto *A*).

Las acciones

Las acciones son representadas por funciones de probabilidad. Las acciones primitivas son:
kick(A,f): si *near(A,b)*, entonces *A* patea la pelota hacia *b* con la fuerza *f* (es decir, en la dirección determinada por las posiciones del jugador y pelota actuales). Esto produce una distancia del pateo, que el jugador puede usar para aprender. Se restringen las distancias que un puntapiés pueden cruzar. No se hace diferencia si el puntapié fue pensado para pasar la pelota a otro jugador o para disparar al arco.
stop(A,M,f): si *near(A,M)*, entonces el jugador *A* intenta reducir la velocidad de *M* con fuerza opuesta *f*. Nos da la velocidad de *M* luego del intento.
move(A,d): el jugador *A* intenta moverse en dirección relativa *d*. Nos da la próxima posición de *A*.

El conjunto de **acciones abstractas** construidas en base a las primitivas y en base a ellas mismas es:
move_to(A,p): el jugador *A* intenta moverse hacia la posición *p* usando una acción *move.* Esto da la próxima posición de *A*.
kick_to(A,d,f): asumiendo que *near(A,b)*, entonces *A* intenta patear la pelota con fuerza *f* en dirección *d*. Se implementa usando *move_to* y *kick*. Da la velocidad y dirección de la pelota.
pass(A,O): asumiendo que *near(A,b)* y *visible(A,O)*, entonces *A* intenta patear la pelota a *O*, lo cual require una llamada *kick_to.*
receive(A,d): El jugador *A* intenta recibir la pelota pateada desde la dirección relativa *d*, lo cual requiere de una llamada a *stop*.

El éxito de las acciones es representado probabilisticamente. De hecho, el jugador no podría estar seguro que la pelota fue pateada en el dirección deseada[2].

[2] En nuestro método, la incertidumbre podría darse debido a *acciones del oponente* o a la *comunicación*. En el primer caso, los oponentes emplearán medidas con el objetivo de crear estados menos deseable para el contrincante (por ejemplo, ellos cambiarán a su defensa para defender bien a un jugador que recibe un pase). Por tal motivo, en la selección de una acción no sólo se deben considerar los datos de los sensores primitivos, sino también los posibles estados futuros.
En el segundo caso, la incertidumbre en la comunicación surge porque cada agente usará una biblioteca de casos diferentes (esto puede verse en detalle en la sección de Razonamiento Basado en Casos).

Las acciones complejas serán guiadas por planes de acción (es decir, una secuencia de acciones) dado que algunas acciones requieren coordinación compleja. Por ejemplo, los planes pueden ser definidos para diferentes líneas de equipos y para diferentes pases (por ejemplo, pases en triángulo, pases por arriba, etc.). La adquisición de un Plan involucra el aprendizaje.
Se usará la acción ***pass*** para ejemplificar el comportamiento del un jugador en un sistema de razonamiento integrado RBC/Bayes (Razonamiento Basado en casos/Bayes).

El Problema

Se combinan dos técnicas: "Bayesiana y Razonamiento Basado en Casos(CBR)" para resolver las tareas que un jugador de fútbol de hacer. Se usa la estrategia Bayesiana ([2] y [3]) para seleccionar acciones y una estrategia de CBR [6] para darle sustento al problema

Seleccionando acciones con Redes Bayesianas

En esta sección, se examina al modelo de acción-selección. Cada jugador usa una Red de Bayesiana ([7] y [8]) para seleccionar una acción. Una red está disponible para cada posible acción. Un jugador evalúa cada red con respecto al estado actual, y selecciona la acción con probabilidad más alta. Este modelo podría aumentarse considerando factores de costo, pero se ignorarán tales detalles en este artículo. Brevemente se describe el contenido, método de la evaluación y componente aprendizaje para nuestras redes de acción Bayesianas.

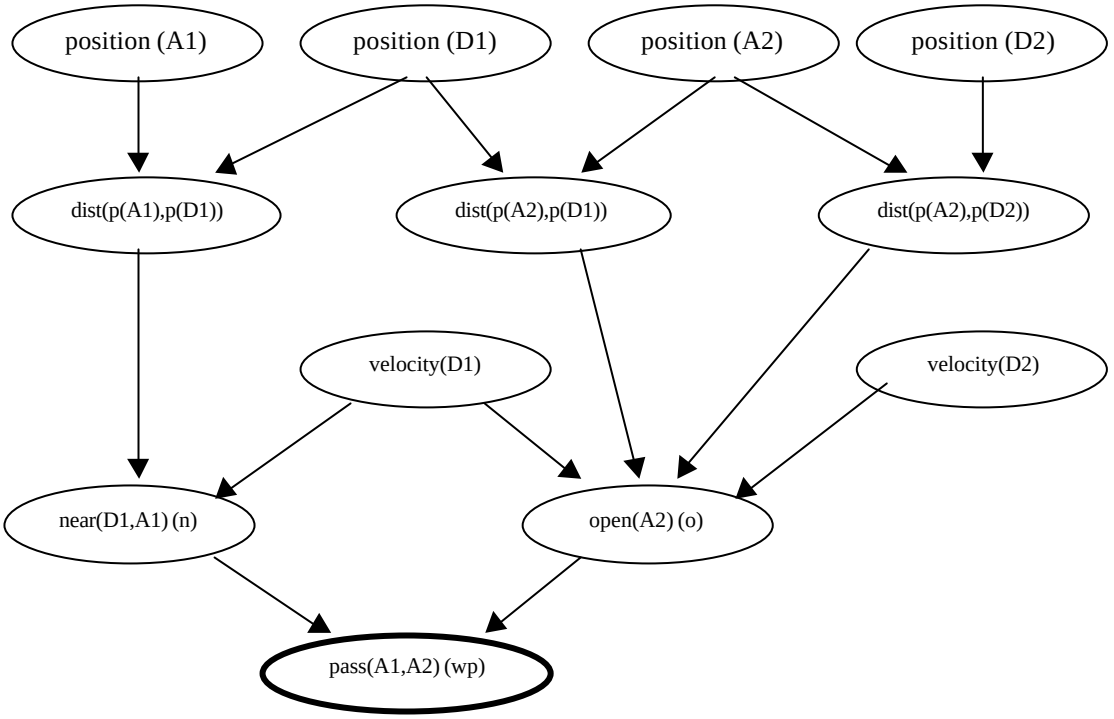


Figura 1.

Figura 1. Modelo de acción para un jugador *A*. El nodo resaltado en negrita es un nodo de acción, los otros son nodos sensores (primitivos o abstractos). *dist* es una abreviación para *distance*, y *p* de *position*.

Contenido: un modelo de acción es una Red de Bayes que contiene nodos sensor y nodos de acción, como muestra la Figura 1. En la misma, los atacantes A1 y A2 son compañeros de equipo, como lo son los defensores D1 y D2. Los nodos sensores representan funciones de sensor primitiva (por ejemplo, *position* y *velocity*) o abstracta (por ejemplo, *distance*, *near* y *open*), mientras que los nodos de acción denotan acciones primitivas o abstractas (por ejemplo, *pass*). Todos estos nodos dan valores Booleanos, y la creencia en sus valores se representa por las probabilidades condicionales. Ejemplo, *pass(A1,A2)* es la acción bajo la consideración de que el jugador A1 debe intentar pasar al pelota al jugador A2.

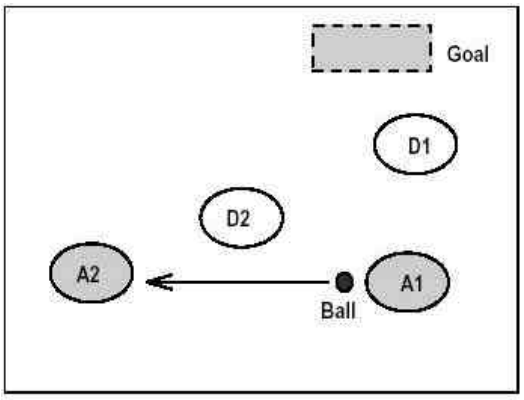
Evaluación: una acción se selecciona computando el grado de creencia (probabilidad de éxito) para cada acción candidata y seleccionando entonces la acción de mayor creencia. Dado que cada acción tiene su propia representación de la red, el método computa el grado de creencia computando las probabilidades condicionales desde la información del estado actual. Por ejemplo, en la Figura 1 se computa la probabilidad de éxito de ***pass*** de la siguente forma:

$$P(pass(A1, A2)) = P(pass(A1, A2) \mid near(D1, A1), open(A2)) \times P(near(D1, A1) \mid d(p(A1)), p(D1)), v(D1)) \times P(open(A2) \mid v(D1), d(p(A2), p(D1)), v(D2), d(p(A2), p(D2)))$$

donde *d*, *p* y *v,* denotan *distance*, *position* y *velocity* respectivamente. Así, la creencia de una acción es igual a la probabilidad posterior dada la información del estado actual, es decir, la evidencia[3].
En el método presente, la conducta del jugador es determinada por la acción seleccionada y la biblioteca del caso de la acción. Nuestro énfasis está en el reuso de planes, dónde los planes son almacenados con los casos.

Aprendizaje (es decir, la actualización de la red): las redes de las acciones son actualizadas cuando (1): una nueva experiencia es adquirida y cuando (2): el conjunto de relevantes características abstractas pertinentes es revisado. Nuevas experiencias son adquiridas intentando ejecutar el plan almacenado de un caso de una acción específica. La calidad del resultado de ejecutar ese plan, junto con el estado resultante, proporciona la regeneración de la red de la acción. Esta regeneración se usa para actualizar las probabilidades condicionales de los enlaces de la red; un algoritmo de actualización podría ser el descripto por Heckerman (1995).

Figura 2.



[3] Se ignora aquí el tema importante de asociar cada acción con su premio potencial, es decir la utilidad de la acción si tiene éxito. Los premios más altos denotan las más valiosas acciones (por ejemplo, marcar un gol).

Un conjunto de nodos pueden ser actualizados. Esto ocurre cuando no todos los pasos de un **plan** de un **caso** recuperado pueden ser ejecutados. Típicamente, los modelos de acción están inicialmente incompletos (por ejemplo, el modelo de acción de Figura 1 está carece de características importantes que involucran al compañero de equipo (*teammate)*, la orientación (*orientation*), etc.).
Considere la escena simulada de la Figura 2. El jugador *A1* "vé" a su compañero de equipo *A2* "libre" (es decir, *open(A2)*) porque computando su creencia dió un alto valor para ¬*near(A2,D2)*. Supongamos que la ejecución de *pass(A1,A2)* falla porque *D2* interceptó la pelota. Este resultado implica que *A1* no consideró la posibilidad de *intercept_threat(D2,A1, position(A2))* cuando analizó la decisión de pasar la pelota a *A2*. Como resultado de esta experiencia negativa, un nodo abstracto respecto a la intersección de *D2* se agregará al modelo de acción en Figura 1. El modelo de acción actualizado se muestra en Figura 3.

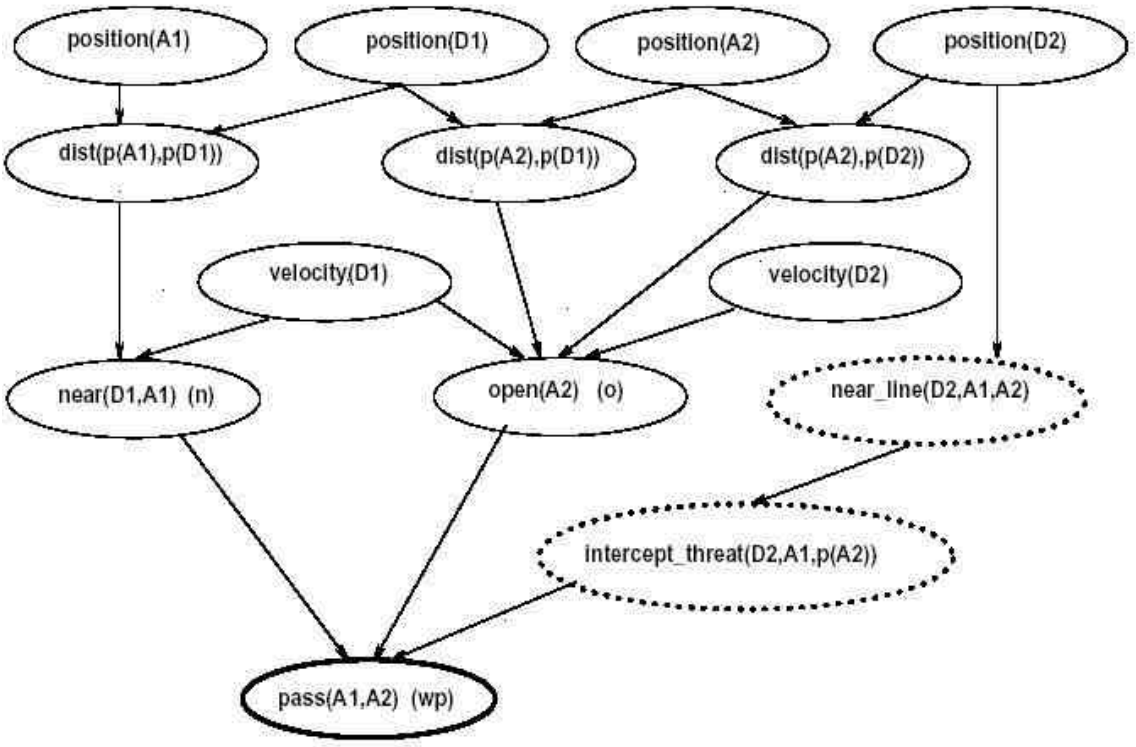


Figura 3.

Como alternativa a este modelo, podría agregarse **peso** a los arcos para reflejar el conocimiento a priori o para reflejar la importancia (peso) del nuevo dato.
Cuando nuevas características se introducen (es decir, en el ejemplo *intercept_threat(D2,A1, position(A2))* la probabilidad a posteriori de la acción candidata debe ser recomputada.

Usando Razonamiento Basado en Casos para Implementar las Acciones

Un caso describe un **plan** [9] para ejecutar una acción seleccionada por un modelo de acción-selección. Dado que una acción puede fallar durante la ejecución los jugadores necesitan reparar la causa de la falla (si información para tal reparación es disponible), o intentar acciones alternativas. Se describe a continuación como se representan, recuperan, reusan, revisan (aprendizaje) y retienen los Casos,

Representación de Casos.

En el ejemplo de la acción *pass(A1, A2)*, *A1* requiere tener cerca la pelota (*near*) y entonces patear esta en una dirección deseada, mientras que *A2* requiere recibir el balón. Definimos así un plan de acción que incluyen tres importantes conceptos: ***condiciones*** (es decir, información derivada de los sensores relevantes), ***secuencia de acciones***, y su ***salida esperada*** (poscondición). Un análisis mas complejo de *pass(A1,A2)* se muestra en la Figura 4 y podemos representar esto de la siguiente manera:

- **Nombre**: *Caso1*
  - *near(A1,b)*
  - *teammate(A1, A2)*
  - *open(A2)*
  - *relative_orientation(b,A2,d)*
- **Secuencia de Acciones**:
  1. *move_to(A1,position(b))*
  2. *kick_to(A1,d,f1)*
  3. *move(A1, NW)*
- **Resultado Esperado**:
  - *receive(A2, -d),* where negation yields the opposite direction.
  - *Possession(A2)*
  - *Kick(A2, N, f2)*

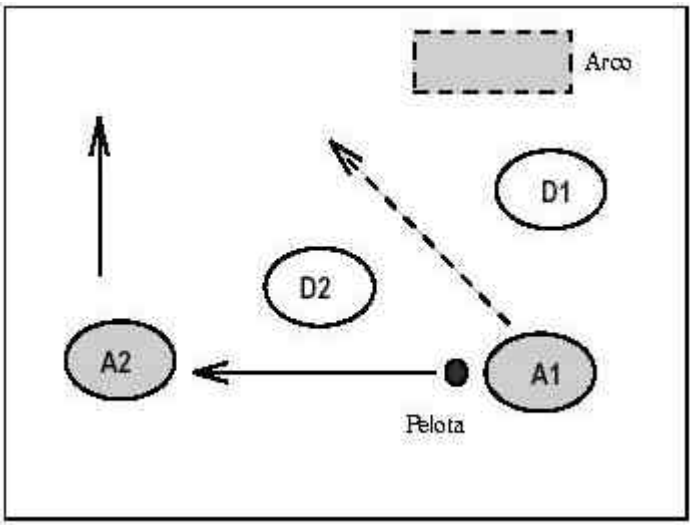


Figura 4.

Recuperación de Casos

La librería de casos es particionada acorde a los planes de acciones de los casos, un subconjunto por cada acción. La similitud es computada sobre aquellos casos que pueden llevar a cabo la acción que ha sido seleccionada por la red bayesiana. Esta similitud es evaluada usando los valores del estado corriente de los sensores que son relevantes a las **condiciones** del caso. En la utilización de los valores de los sensores hay que tener en cuenta el grado de error que estos pueden tener. El caso mas similar es recuperado.

Reuso de Casos

Cuando un Caso es recuperado, su plan de acción se activa. Primero de todo, sensores especificos y nodos característicos de la red bayesiana son medidos, lo cual es determinado por las condiciones del caso. Luego las acciones del caso son ejecutadas. Una acción no primitiva, producirá mas recuperaciones de casos, requiriendo así de su propia librería de casos.

Revisión de Casos

La pasos de acción de un Caso son ejecutados en secuencia hasta que la secuencia es completada o hasta un paso de acción no puede ser llevado a cabo, lo cual es una señal para una revisión del caso. La reparación de la falla, la cual puede involucrar una considerable evaluación, será postergada hasta después de la ejecución.
Las posibles causas para una falla de ejecución conciernen: al modelo de selección de acción, la librería de casos y/o a una información imprecisa de los sensores. En la presente sección se describe una ejemplo donde el modelo de selección de acción fue revisado para tratar de asegurar un pase (*pass*) sin que ningún oponente lo intercepte. Esta es una revisión de falla general, que se aplica a todos los casos de su librería de casos asociada. Cuando la falla no es relevante para todos los casos de una acción dada, las revisiones son hechas al caso seleccionado y no toda la librería. Las revisiones pueden tener efecto sobre cualquiera de los siguientes tres escenarios: sobre las condiciones, sobre la secuencia de acción y/o sobre la salida esperada (poscondición), cualquiera de estos tres tipos de revisiones intentan hacer que el jugador pueda completar con éxito la acción.
Las condiciones de un caso puede ser modificada cuando la falla sugiere que la secuencia de acciones hubiese tenido éxito si una cierta modificación se hubiese efectuado. Un seteo de parámetro incorrecto, tal como la fuerza del pase produce una simple modificación, pero un ejemplo un poco mas complicado es el siguiente: si el campo de juego se encuentra mojado por alguna lluvia reciente, una acción correspondiente al intento de anotar un gol deberia ser mejor un pase por el aire mas que por el suelo. Una condición de caso revisada, debería ahora incluir un sensor primitivo como *campo_mojado()*. Esto también produce un segundo caso correspondiente a la condición ¬ *campo_mojado()*.
Otro ejemplo que extiende la Figura 4, en el cual si la creencia (probabilidad a posteriori) de que *D2* intercepte la pelota es alta (*intercept_threat(D2,A1, position(A2))*, entonces la pelote debe patearse sobre un costado de *A2* tal que su dirección se aleje de *D2*. Este escenario se ilustra en la Figura 5.

Los cambio entonces que se producen en Caso1, el cual llamaremos Caso2, son los siguientes:

- **Nombre**: *Caso2*
  - *near(A1,b)*
  - *teammate(A1, A2)*
  - (*intercept_threat(D,A1, position(A2))*
  - *open(p)*
  - *relative_orientation(b,p,d)*
  - *near(A2,p)*
- **Secuencia de Acciones**:
  1. *move_to(A1,position(b))*
  2. *kick_to(A1,d,f1)*
  3. *move(A1, NW)*
- **Resultado Esperado**:
  - *move(A2, p)*
  - *receive(A2, -d)*
  - *Possession(A2)*
  - *Kick(A2, N, f2)*

Como se puede observar, este mismo ejemplo muestra la tercer manera que un Caso puede ser modificado, es decir, modificando su salida esperada. Notar que el jugador *A1*, asume que *A2* se moverá hacia la nueva posición donde la pelota fue pateada, esto tiene mucho que ver con la **cooperación entre jugadores** que será discutida mas adelante.

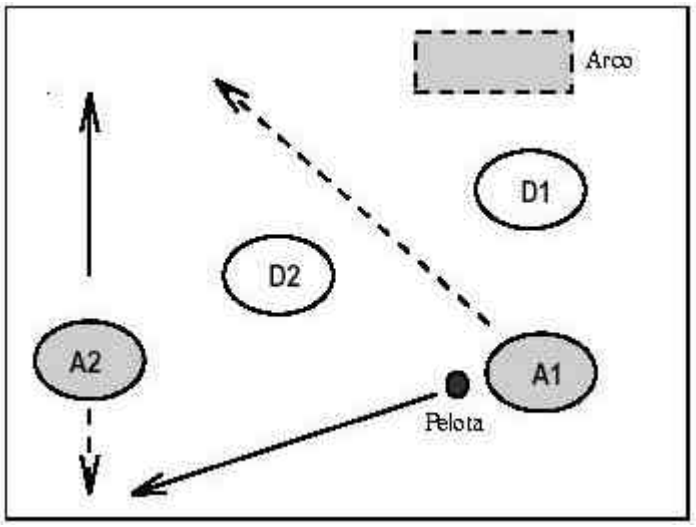


Figura 5.

Retención de Casos

La retención involucra actualizaciones sobre las librerías de casos. Continuando con nuestro ejemplo, se guardarían dos casos por el fallo de la ejecución causado por la interceptación de D2. El primer caso corresponde a la situación dónde (*intercept_threat(D,A1, position(A2))* es True, lo cual produce el **Case2** como se mostró anteriormente. El segundo caso es una extensión de **Case1** con la condición adicional que ningún oponente intercepte la pelota. Ambos casos se guardan para cubrir bien el espacio de posibles estados.

Cooperación entre Jugadores

En esta sección, se discuten las estrategias para la cooperación del jugador. Aunque cada jugador aprende de sus propias experiencias, se asumen que los resultados sabios son compartido entre todos los jugadores. Sin embargo, dado que las capacidades de percepción de cada jugador está limitada, los casos que ellos seleccionan serán diferentes, y esto puede llevar a las preocupaciones sobre cómo animar la cooperación inter-equipo.
Así, varios problemas necesitan ser tratados cuidadosamente al seleccionar una estrategia alentadora para la cooperación del inter-equipo. Dos de éstos problemas más cruciales son:
1. ***intención***: Un agente quiere sólo comunicar su intención a sus compañeros de equipo. Se espera que la información comunicada a sus oponentes sea menos útil o engañosa.
2. ***anticipación***: los jgadores deben anticiparse a las acciones de sus compañeros de equipo para producir una correcta y planificada cooperación. Cada agente tiene tendencias individuales, y el trabajo en equipo involucra el aprendizaje de cómo mejorar la anticipación del comportamiento de cada compañero de equipo.

Se exploran en esta área dos acercamientos, ***"compartido"*** e ***"individual"***, para la cooperación alentadora entre agentes del mismo equipo. El primero trata sobre estrategias de control *compartido*, en la cual un director centralizado entra la acción seleccionada e instancia un set de jugadores involucrados en el plan. Esto es algo análogo a la idea de aplicar estrategias pre-planeadas que son dadas a por un Técnico de equipo, las cuales dicen lo que se planea seguir antes de actuar a jugador. Así, las preocupaciones de ***intención*** y ***anticipación*** son bastantes simplificadas.
Por el contrario, la estrategia del control individual es más ambiciosa ya que exige a cada jugador seleccionar su propia acción, sin la información de las acciones seleccionadas de sus compañeros de equipo. Así, las preocupaciones con respecto a la intención y anticipación son grandes. Desde que los planes de acción tienen las expectativas (salida esperada) explícitas acerca de la conducta de los jugadores, los jugadores pueden trabajar juntos si está disponible el set de casos que comparten las mismas condiciones. Por ejemplo, suponga A2 selecciona el caso Case3 de acuerdo con su plan. Las condiciones de Case3 son "similares" a las del Case1 (un caso correspondiente a la librería de casos de A1), entonces es de esperar para A1 que A2 tenga las mismas expectativas que el Caso1 (pero hay que tener en cuenta que el Caso1 hace referencia a A1), con lo cual puede anticiparse a las acciones de A2.

Conclusiones

En este artículo, se propone resolver el problema del juego de fútbol con varios jugadores que planean sus tareas. El método usa redes Bayesianas para seleccionar acciones y el razonamiento basado en Casos para llevar a cabo las acciones seleccionadas. Sería muy bueno usar el simulador de fútbol RoboCup como un testeador del método presentado. Esto involucraría las siguiente tres estapas:

1. Aumentar el software de RoboCup para extender las capacidades del jugador (por ejemplo, el juego de sensores primitivos disponibles).
2. Construir los componentes del Razonamiento Basado en Casos (por ejemplo, memoria estructurada) y una interface para la red Bayesiana

3. Evalúe el método integrado, en las competiciones de equipo de fútbol.

Referencias

[1] Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications, 7, 39-59.

[2] Jelle Kok, Remco de Boer, Nikos Vlassis, and Frans Groen. "Towards an Optimal Scoring Policy for Simulated Soccer Agents". Intelligent Autonomous Systems Group, Informatics Institute Faculty of Science, University of Amsterdam Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

[3] Georgios Koudouridis 1999. "Learning and Assigning Probabilities to Decision Trees: Soccer Case Study".Course Agent Programming II.

[4] GZ Yang and DF Gillies ."Bayesian Inference in Vision". Computer Vision © Department of Computing, Imperial College.sds. http://www.doc.ic.ac.uk/~gzy

[5] Jhon James Mora. Apunte Introductorio de la Técnica Bayesiana.

[6] Breese, J. S., & Heckerman, D. (1995). Decision-theoretic case-based reasoning. Proceedings of the Fifth International Workshop on Articial Intelligence and Statistics (pp. 56-63). Ft. Lauderdale, FL: Unpublished.

[7] Cooper,G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. Articial Intelligence, 42, 393-405.

[8] Heckerman, D. (1995). A tutorial on learning Bayesian networks (Tecnical Report MSR-TR-95-06). Redmond, WA: Microsoft Corporation, Advanced Technology Division, Microsoft Research.

[9] Kushmerick, N., Hanks, S., & Weld, D. (1994). An algorithm for probabilistic least-commitment planning. In Proceedings of the Twelfth National Conference on Arti_cial Intelligence (pp.1073{1078). Seattle, WA: Morgan Kaufmann.

[10] Apunte elaborado en la "Escuela Técnica Superior de Ingenieros de Telecomunicaciones", Universidad de Valladolid.